ERDS: Emerging Risks Detection Support

2007 project report

Lars Hulzebos
lars.hulzebos@wur.nl

Jeen Broekstra
jeen.broekstra@wur.nl

The research reported here has been carried out in 2007 as part of the BO-08-005 project ”Emerging Risks in de Nederlandse Voedselketen”. This project is funded by the Dutch ministry of Agriculture (LNV) and supervised by the Dutch Food Safety Authority (VWA).

Agrotechnology & Food Innovations B.V.
P.O. Box 17
6700 AA Wageningen
The Netherlands
tel. +31 (0)317 475 029

Contents

Chapter 1

Introduction

This report describes the activities and results on the development of a prototype for the detection of emerging risks in the fish chain. This effort was part of the BO-08-005 project in 2007 in which VWA, RIKILT, IMARES, AFSG, LEI and PRI participate. BO-08-005 is funded by the dutch ministry of agriculture (LNV) and supervised by the dutch food safety authority (VWA) [Marvin, 2008].

In order to define emerging risk detection, we have to introduce the areas of attention in food safety (see Figure 1.1):

- *food safety research*[1], is about creating new scientific knowledge and re-evaluating existing knowledge. Research can be given a certain direction by knowledge questions emerging from the other areas of attention.
- *emerging risks detection*, focusses on detecting risks related to food consumption that will emerge in the near or far future. The process of detecting requires the state-of-the-art of food safety research and knowledge on how food is processed, where it is obtained from, how it is processed *et cetera*.
- *early warning detection*, is the process of continuously monitoring parameters in food (related) chains and looking for violations of (legislative) thresholds. Emerging Risks detection can imply a change in the monitoring plan like the parameters that are monitored, the frequency of sampling and even the method of sampling.
- *risk communication*, is the process of dealing with an exception or a crisis. It is crucial to take the right measures when a treshold is violated in or before the chain-link in the food (related) chain.

This report focuses on emerging risk detection. Without emerging risk detection, risks can emerge in the near or far future without being noticed (See Figure 1.2). When a crisis actually breaks out, one can only afterwards see the signals/events that lead to the emerged risk. Emerging risks detection tries to identify emerging risks based on current and historical data. Those predicted emerging risks can give stakeholders time to control the predicted risk by taking the right measures.

[1] the emphasized terms are terms firstly introduces and containing a definition. They also appear in the index chapter

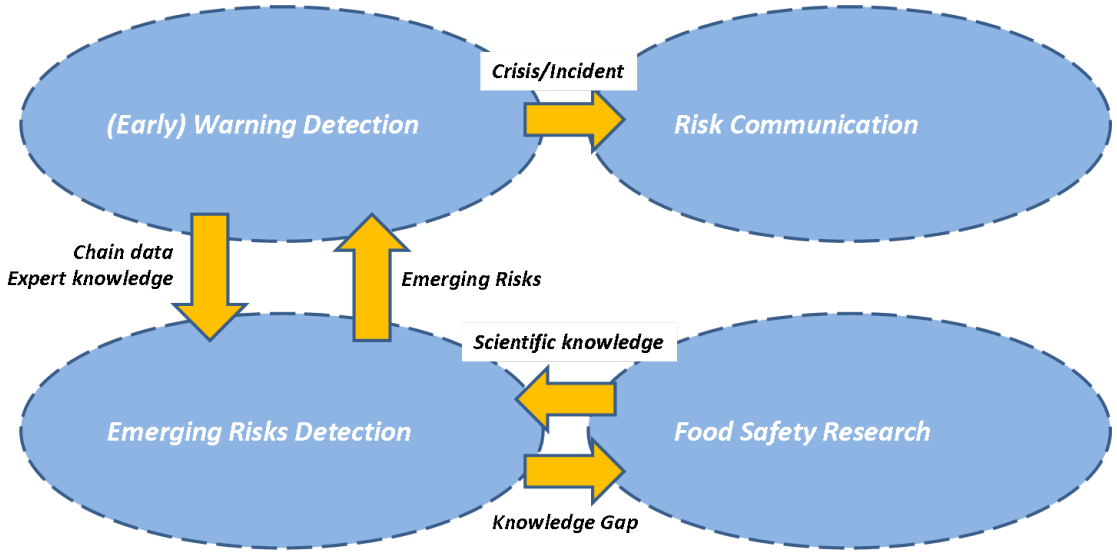

Figure 1.1: An overview picture in PNG format

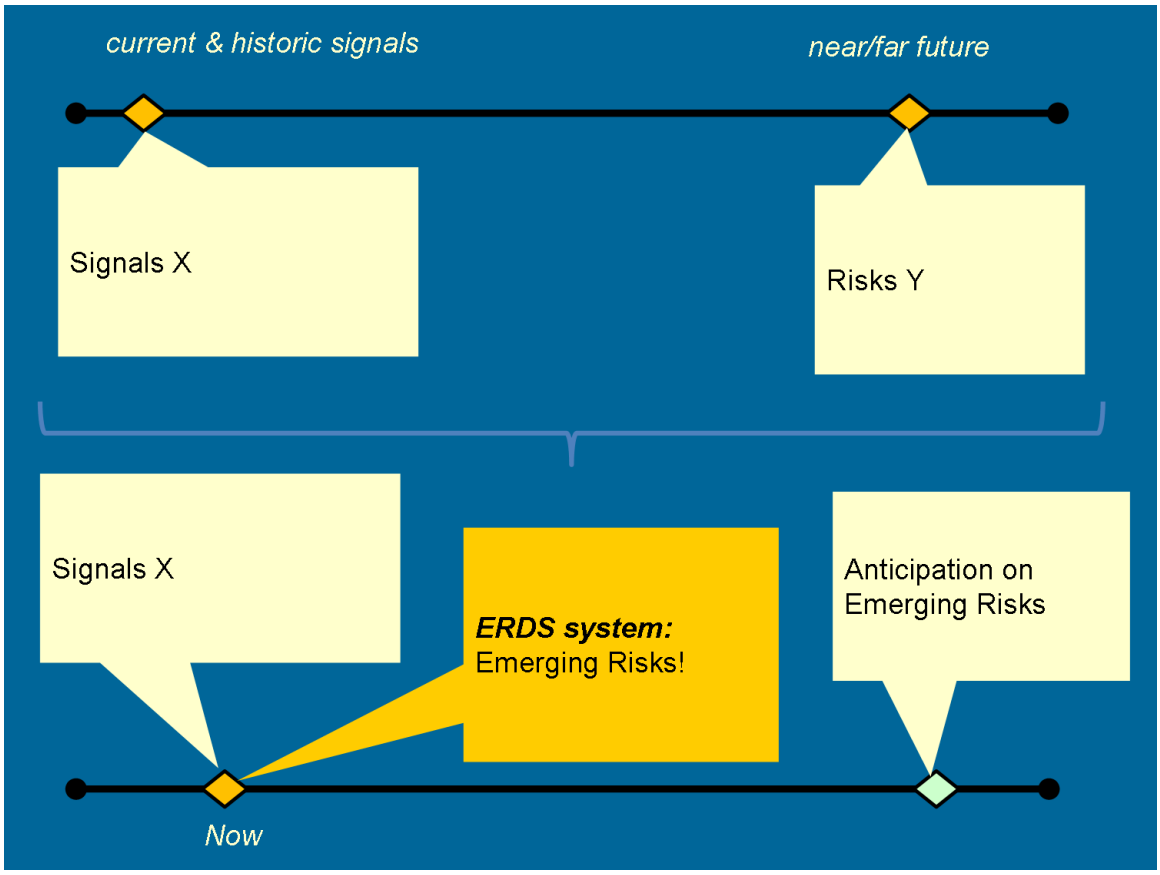

Figure 1.2: Situation without detecting emerging risks (above), and with detecting emering risks (below).

Chapter 2

The Melamine Case

2.1 introduction

In 2007 we focused on one actual case that illustrates the power of detecting emerging risk. We came up with the pet-food crises that emerged during Late 2006 and early 2007 onwards. In that period pets in the USA became ill and died after consuming a certain brand of pet food. After investigation it turned out that the wheat used in the pet food was contaminated with Melamine (an organic compound leading to kidney failure). The contaminated wheat was imported from a company in China. On the 30th of March the FDA[1] blocked the import of all products from this particular company. From the pet feed industry point-of-view this case was closed. However on the 9th of May 2007 Melamine was also found in hatched salmon in Canada.

We claim that this finding outside the pet feed chain could have been foreseen by using an emerging risk detection system. Such a system needs the following knowledge describing:

- the involved companies, used and produced products
- the food safety expert knowledge on how risks can emerge and propagate
- the signals that authorities give out like the banning of wheat from a chinese company by the FDA

With this knowledge, such a system could have detected an emerging risk for the hatched salmon in Canada as early as the 30th of March 2007. (see Figure 2.1).

2.2 the detection of emerging risks

Detecting emerging risks is about combining chain- and expert knowledge with *signals*. A signal represent a fact, state or event in the real world. The signal is seen as a relevant *indicator* on an emerging risk in food chains [Noteborn, 2006]. Signals are derived from a data source. A *data source* is a digital source like web site or electronic document containing one or more signals. An example of such a signal is 'the contamination of wheat in the USA with Melamine'. The knowledge that we gathered from experts in the field of food safety and food chains was:

- Bulkproducts are often distributed to all countries of a continent where the country lies in which the bulkproduct is present
- If a hazard on a product in a food chain is present, that hazard applies in most cases to the rest of the (downstream) chain

[1] Food and Drug Authority in the USA, is an agency of the United States Department of Health and Human Services and is responsible for the safety regulation of most types of foods, dietary supplements, drugs, vaccines, biological medical products, blood products, medical devices, radiation-emitting devices, veterinary products, and cosmetics.

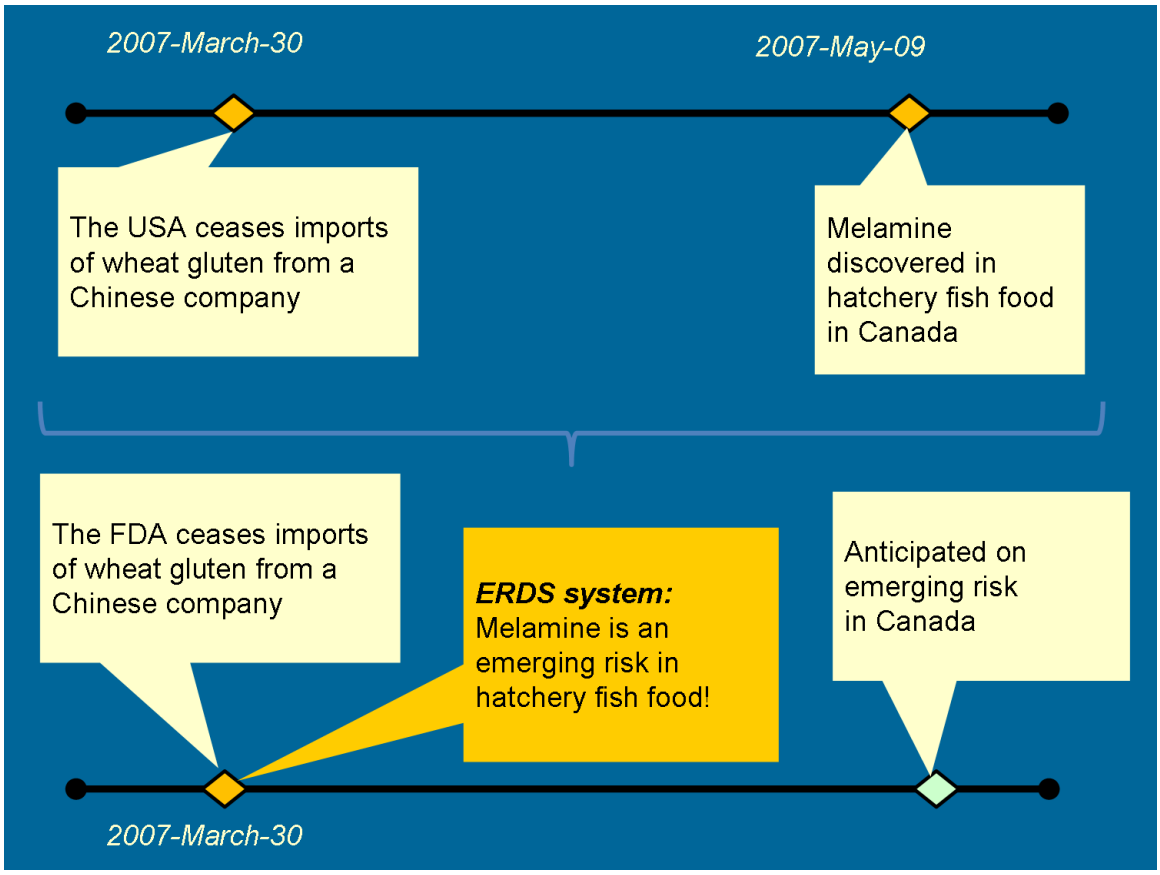

Figure 2.1: Situation without detecting emerging risks (above), and with detecting emering risks (below).

- If a food stuff is contaminated with an organic contaminant, one speaks of a chemical hazard
- Melamine is an organic contaminant
- An organic contaminant on food is a health hazard
- A hazard is a biological, chemical or physical agent in, or condition of food or feed with the potential to cause an adverse health effect.[Noteborn, 2006]
- A chemical hazard is a health hazard
- Wheat is a kind of food, also a bulk product and is used in fish feed
- Hatched fish is fed with fish feed and farmed in Canada

To make the detection of emerging risks as manageable as possible it is essential to focus on the type of risks of interest. In the melamine case we are interested in fish with a health risk for consumers in the Netherlands[2]. This risk of interest is the *goal definition*. Any route from a signal via expert knowledge to the goal definition is called a *risk pathway*.

Figure 2.2 shows the start situation of the system. The red-crosses (right below) represent the type of risk the user is interested in. The green-crosses (left-up) represent one of the signals the system retrieved from a data source. In this case the signal represents the contamination of wheat in the usa. This fact is

[2]assuming that dutch governmental organisations are the end-user of the system

derived from the banning report on the FDA site on the 30th of March 2007.

The system uses its chain- and expert knowledge to derive as much other facts (or *derived facts*) from the previous known signals. The expert knowledge is split into many small but comprehensive *knowledge fragments*. These knowledge fragments Wheat is a kind of food, Melamine is an organic compound and Organic contaminant on food is a health hazard derives the new fact Wheat contaminated with melamine is a health hazard. This is depicted in Figure 2.3.

Combining this with the knowledge fragments Bulk products available in north America is available in each country of north America and Wheat is a bulk product, derives the fact Wheat contaminated with Melamine that is a health hazard is present in Canada See Figure 2.4.

Using the knowledge fragments Fish farmers that breed salon are located in Canada, Salmon is fed with fish feed and Wheat is a fish feed in combination with the already derived previous fact results in the conclusion A Fish farmer in Canada is using potentially contaminated wheat with melamine which is a health hazard (Figure 2.5.

The fact Fish is located in Canada contaminated with Melamine is derived, as depicted in Figure 2.6 from the fragments Organic contaminants remain in all derived products in the food chain, Fish farmers breeding salmon are located in Canada and Salmon is a kind of fish combined with the already derived fact A Fish farmer in Canada is using potentially contaminated wheat with melamine.

Using the fragments An organic contaminant on food is a health hazard, together with Fish is a kind of food on the derived fact Fish is located in Canada contaminated with Melamine expands this fact to Fish is located in Canada contaminated with Melamine which is a health hazard (depicted in Figure 2.7.

Figure 2.8 uses the fragment Exporters from salmon are located in Canada to conclude that Exporters in Canada trade in Fish contaminated with Melamine that is a health hazard by combining it with the previous derived fact.

Finally (Figure 2.9) makes the link to Europe by applying the fragments importers from salmon are located in Europe, and Salmon is transported from Canada to Europe with previous derived fact Exporters in Canada trade in Fish contaminated with Melamine that is a health hazard to the fact In Europe Salmon contaminated with melamine which is a health hazard is imported. This derived fact intersects with the goal of a health hazard on Fish in Europe, meaning that a risk pathway is found (Figure 2.10).

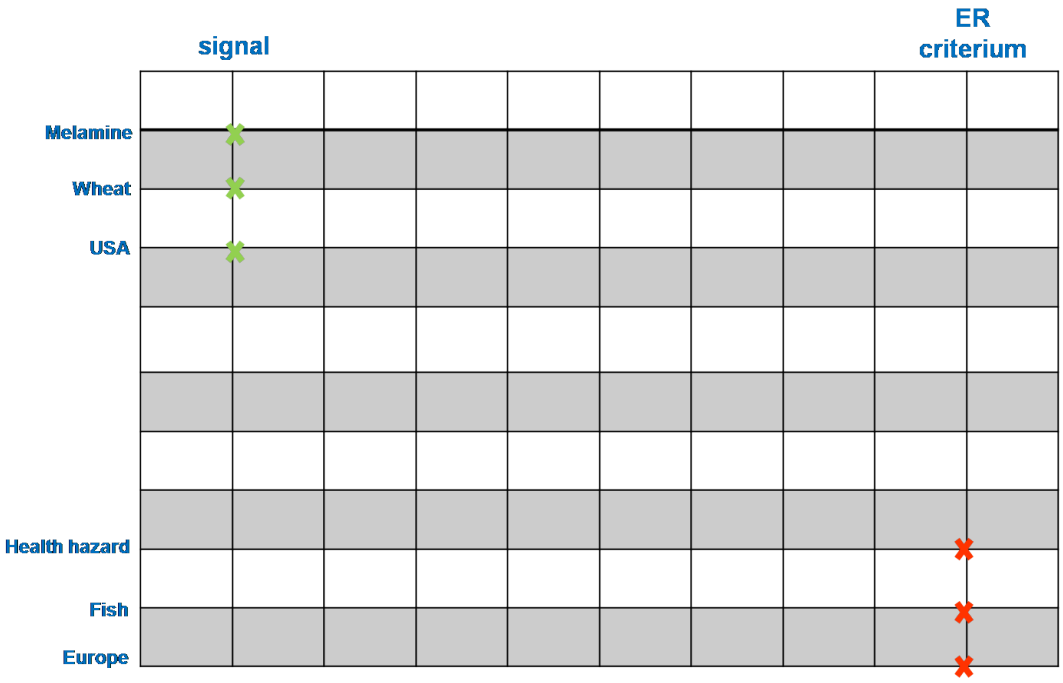

Figure 2.2: Finding a risk-pathway 1

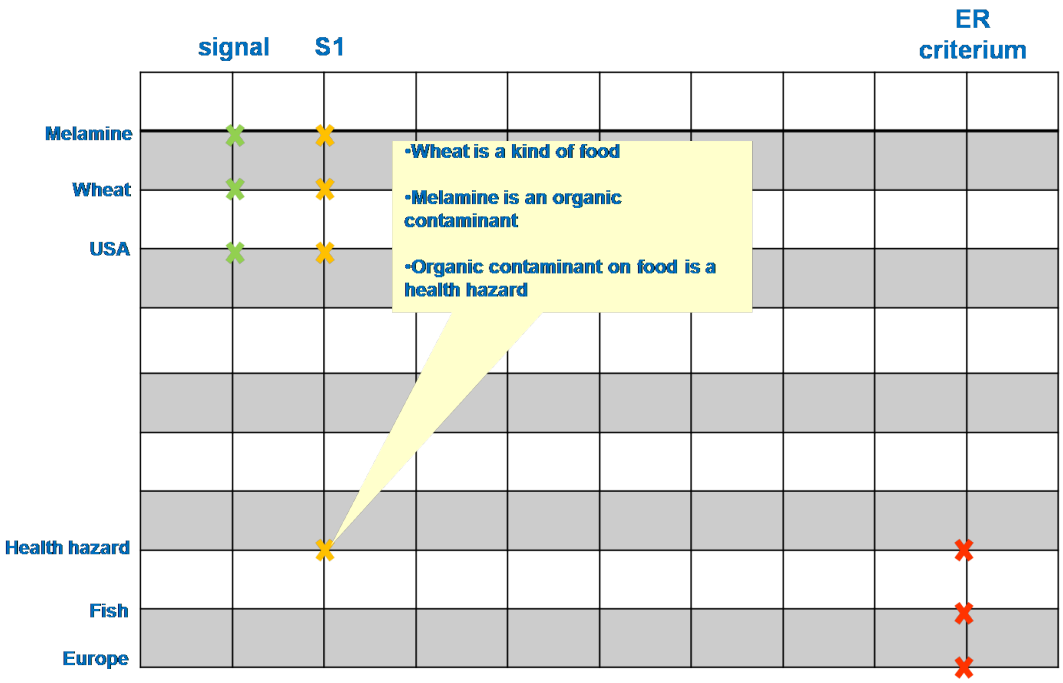

Figure 2.3: Finding a risk-pathway 2

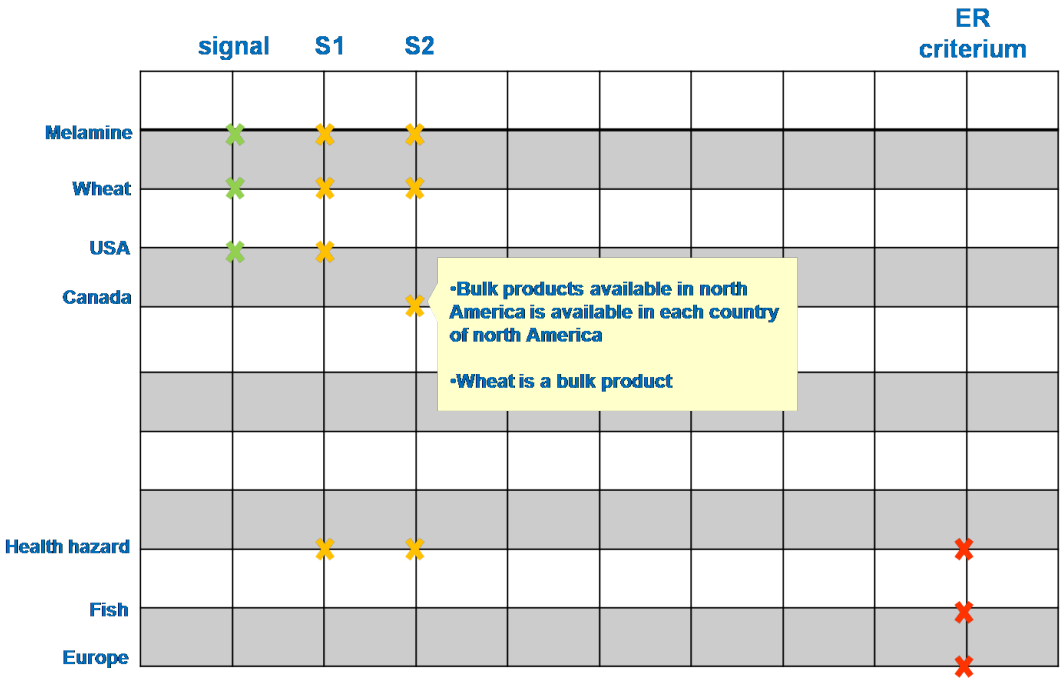

Figure 2.4: Finding a risk-pathway 3

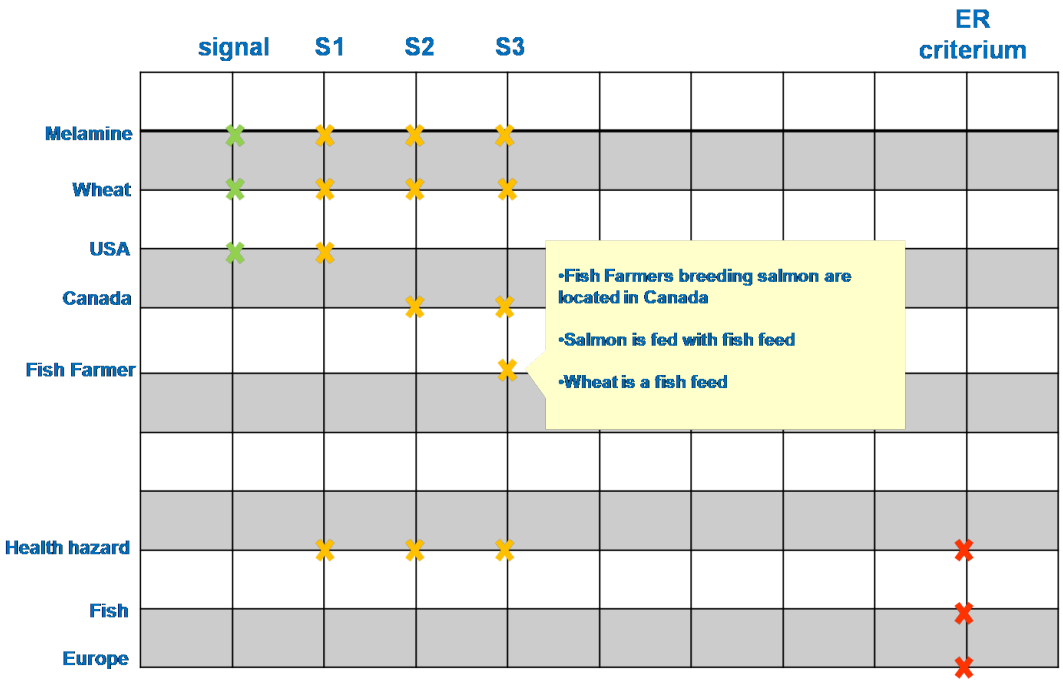

Figure 2.5: Finding a risk-pathway 4

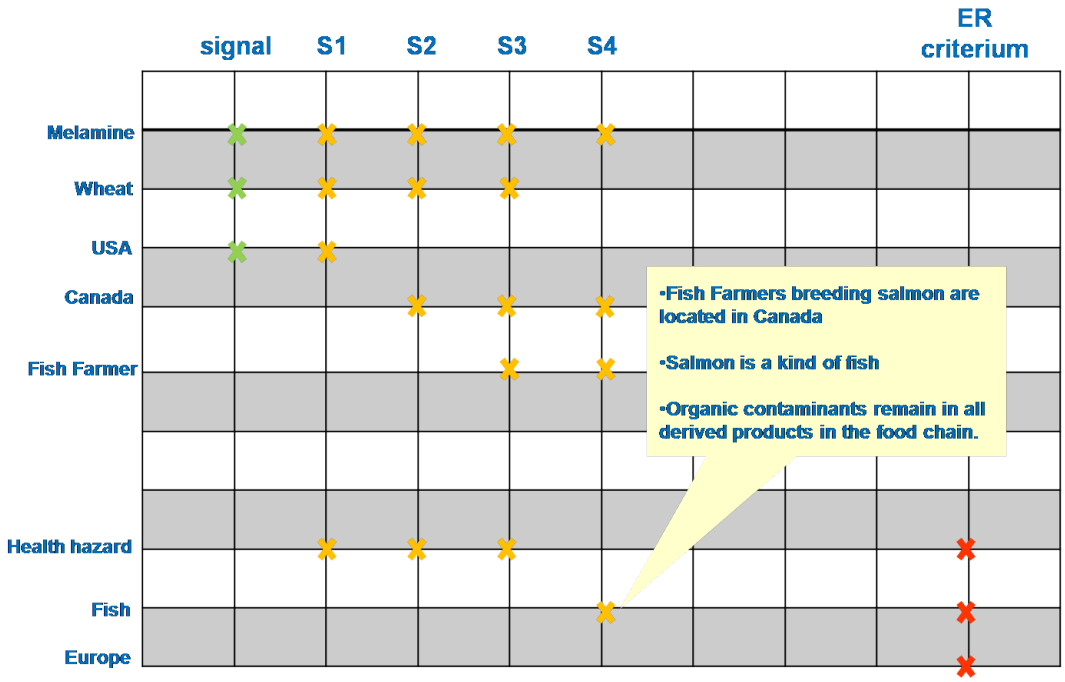

Figure 2.6: Finding a risk-pathway 5

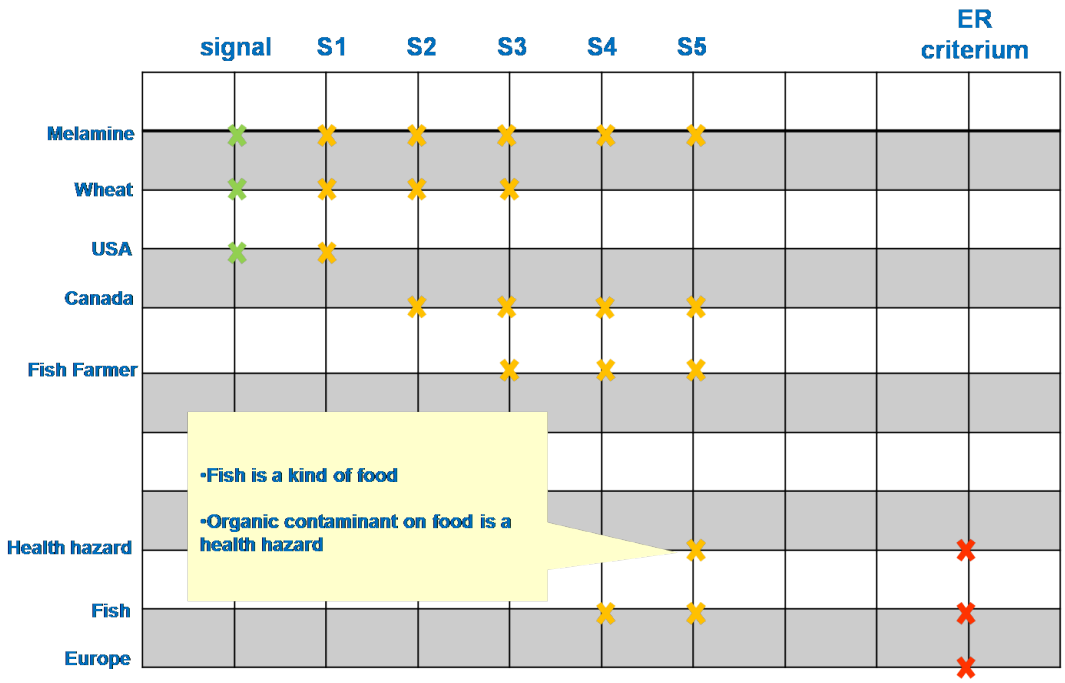

Figure 2.7: Finding a risk-pathway 6

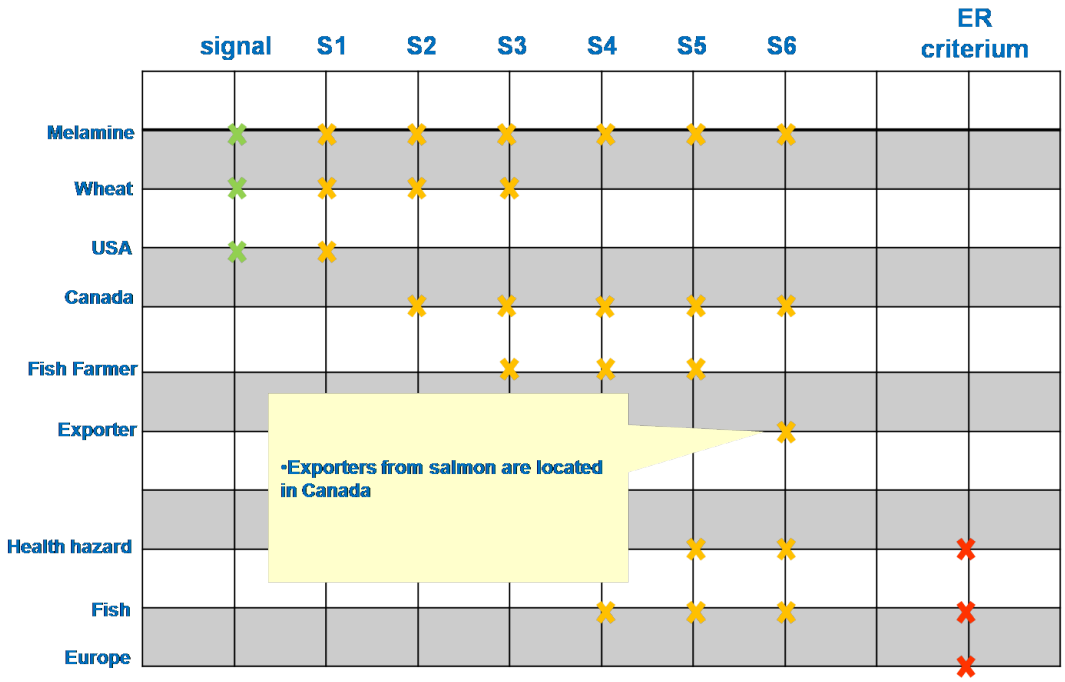

Figure 2.8: Finding a risk-pathway 7

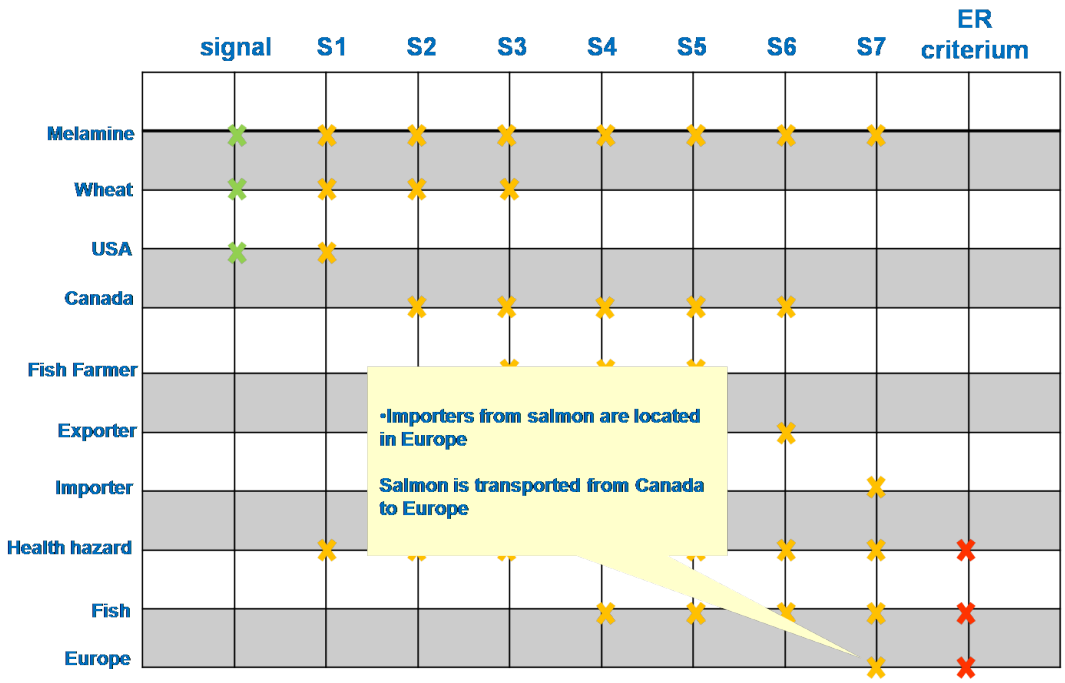

Figure 2.9: Finding a risk-pathway 8

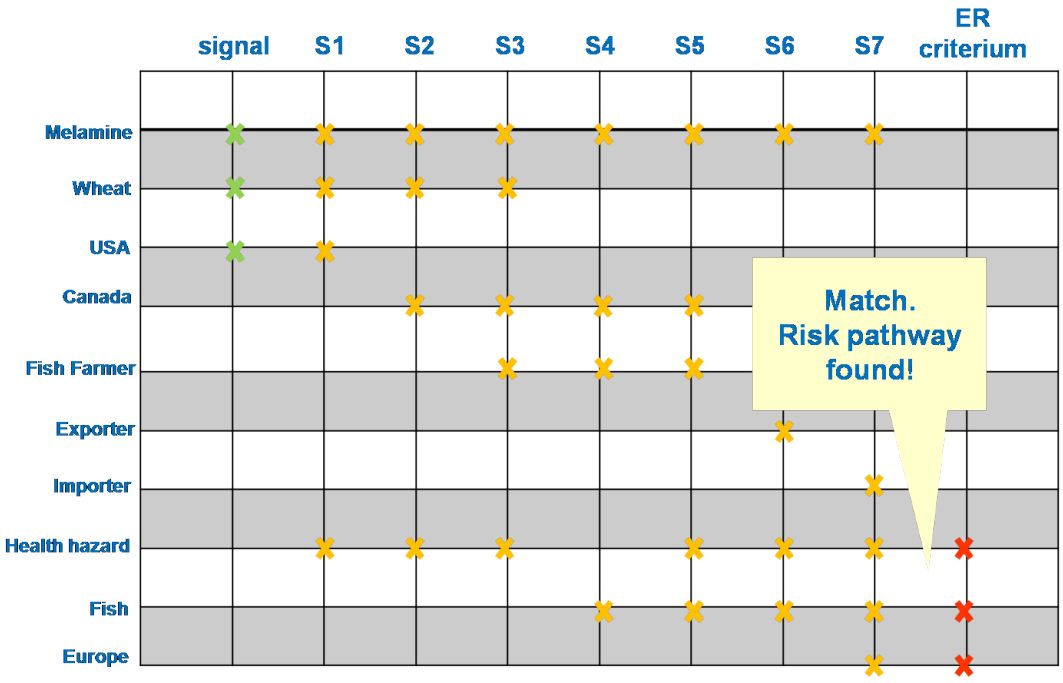

Figure 2.10: Finding a risk-pathway 9

Chapter 3

The ERDS Workflow & Scenarios

3.1 End Users

3.1.1 Introduction

We see the food safety agencies as the end-user of the ERDS System. Those agencies are mandaded by the policy maker to carry out and control food safety monitoring and have operational and practical experiences in food chains. This makes the understanding of the ERDS System's output for this target audience relatively easy. (See Figure 3.1)
From the industry point-of-view we initially saw no direct end-user of the ERDS concept, since the daily operations in the industry is short-term focused. However in some food sectors the industry itself initiated some obligatory international and national quality standards like BRC[1] and IKB[2] using it also as a marketing instrument to make a distinction on quality among competitors. These quality standards focuses more on the long-term. The ERDS System can improve such a quality standard by identifying emerging risks. The quality standard can adjust their criteria to deal with that particular emerging risk, even before it is obliged by law.

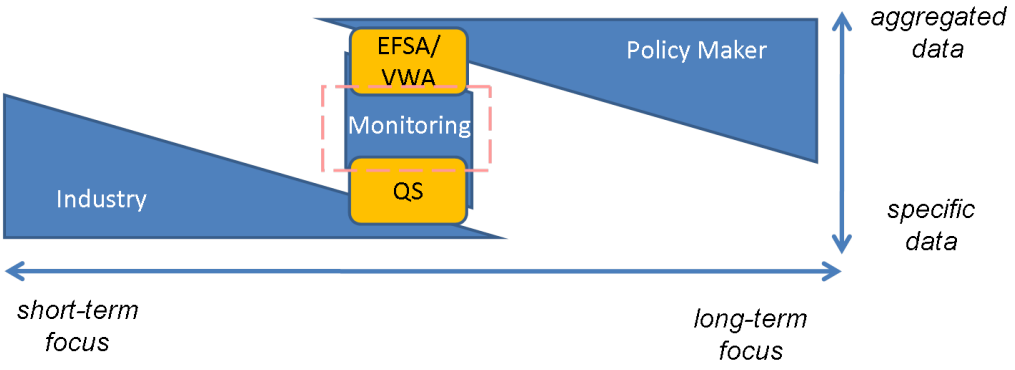

Figure 3.1: Potential end-users of the ERDS system

3.1.2 End-User's Workflow

In Figure 3.2 the functionality and workflow of the ERDS System is depicted. After entering the application the user can either directly detect emerging risks or firstly review and/or add new facts by himself. Note

[1] http://www.brc.org.uk
[2] IKB varken: http://www.ikbvarken.nl IKB 2004: http://www.dgbbv.nl/ikb2004

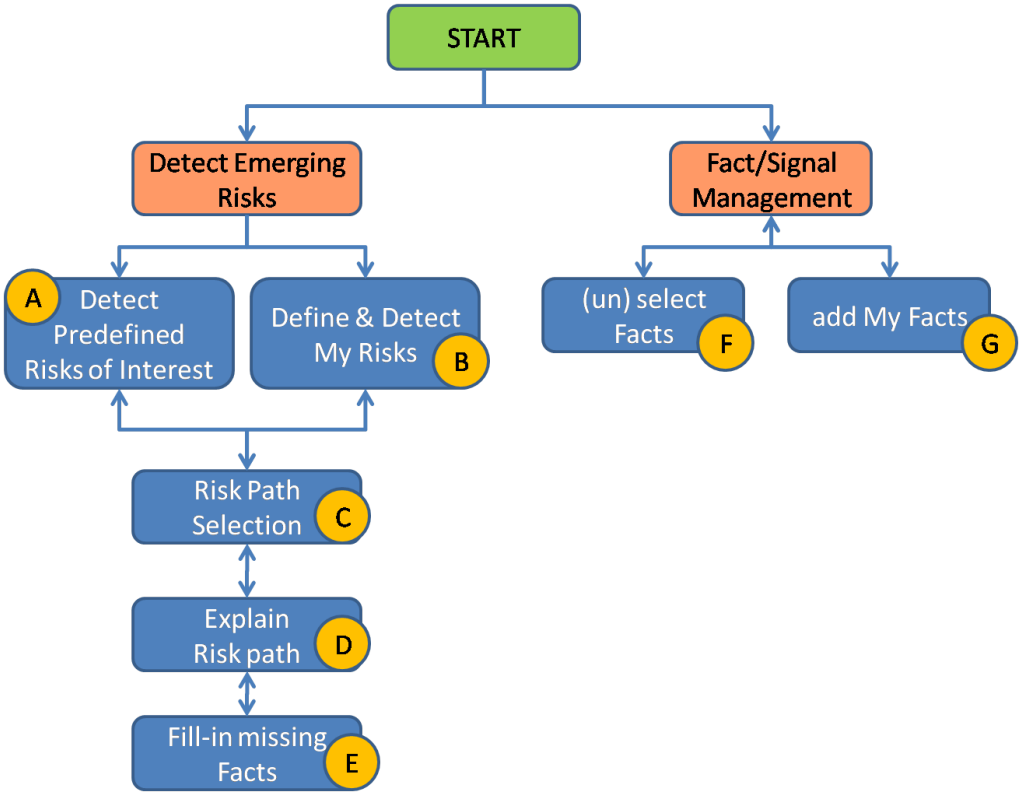

Figure 3.2: The workflow for end-users of the ERDS system

that these added facts are socalled 'play' facts and used for a what-if scenario play. A what-if scenario is about getting a feeling of the effects on assumed causes. Like 'Is there a risk pathway to fish consumption in the Netherlands if wheat in Mexico was contaminated with Melamine?'. When the user wants to detect emerging risks he can either see the found risk-pathways of predefined areas of attention (like emerging risks on fish, meat, vegetables or fruit in the Netherlands). In Figure 3.3 one can see that for the area of attention fish three risk pathways are found that lead to a high emerging risk, one to a medium emerging risk and zero to a low emerging risk.[3]

When the user choose to define the goal criteria of a risk pathway by himself he goes to the screen as depicted in Figure 3.4. In this case he is interested in all emerging risk pathways that regards fish products in europe with a health hazard.

When the system searches for either a predefined or a manual entered riskpathway it responds to the user as can be seen in Figure 3.5. One can see that two lists of found pathways are shown: The pathways where all information (i.e. signals) are complete (the top-list to the left), and the pathways where one or more signals are missing. The latter list is shown left-below.

Once the user has selected a pathway from one of the lists, the pathway itself is shown on the right (see Figure 3.6). The squares represent a (derived) fact and the arrows represent the expert knowledge. In this case the user has selected one of the arrows to receive information on the reasoning-step that the system has made to derive a fact from a previous one.

[3]This screenshot is a mock-up. Until now we have not addressed the quantification of an emerging risk. The question when a pathway is a low, medium or high risk still remains. We now only take 'risk' into account based on expert knowledge.

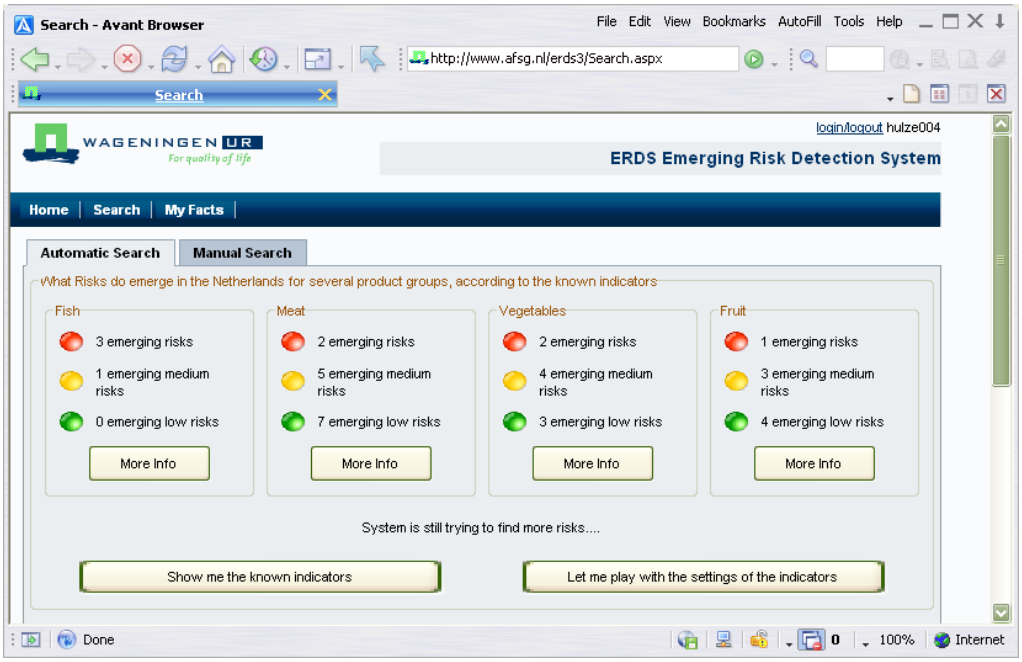

Figure 3.3: Detecting predefined risks of interest (A)

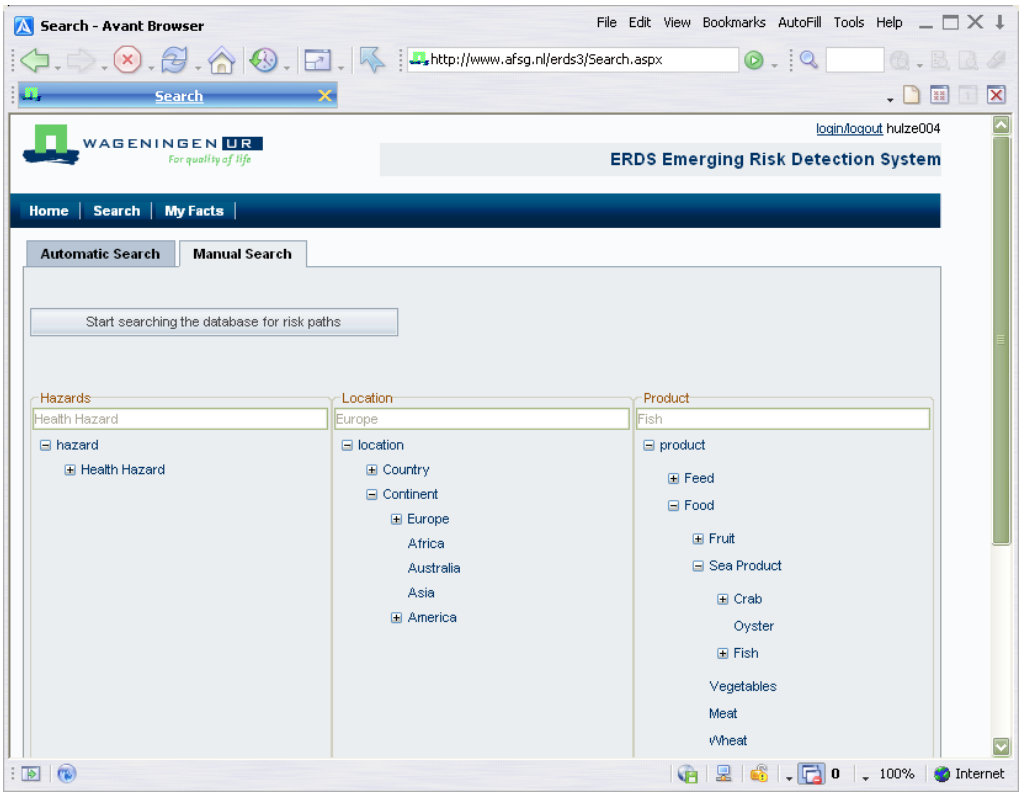

Figure 3.4: Define and detect custom made risks (my risks) (B)

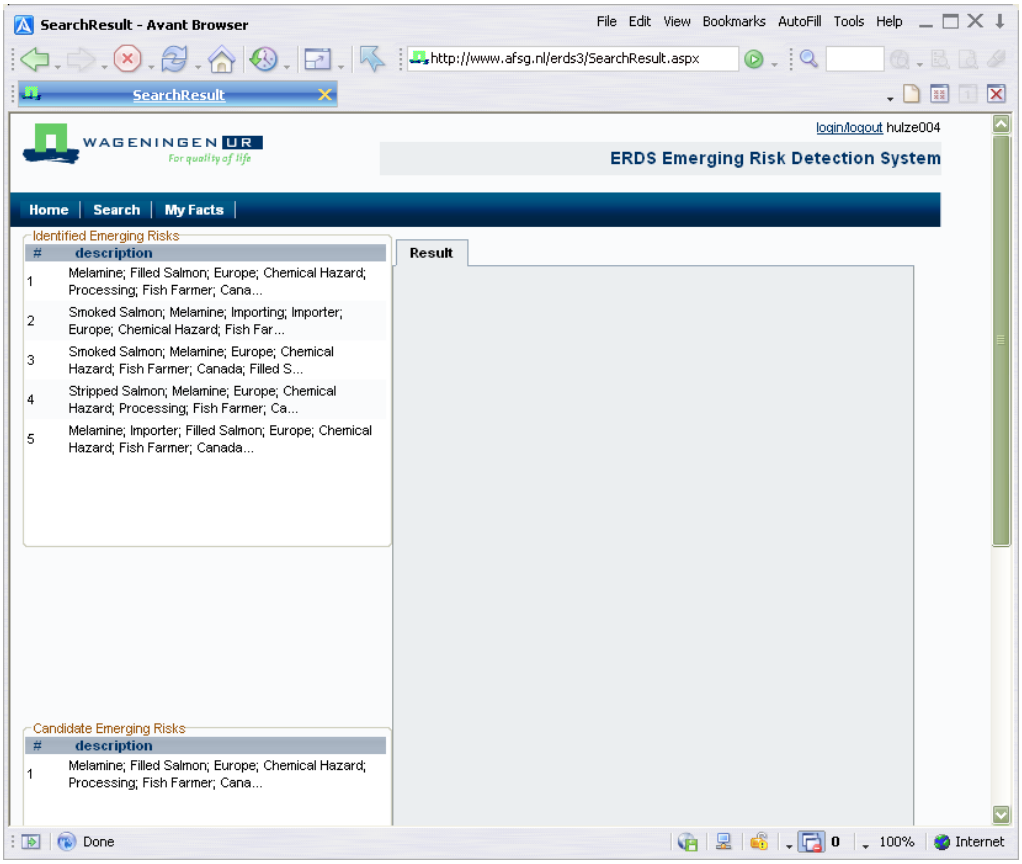

Figure 3.5: Risk path selection (C)

In the selected risk-path one indicator was not known to the system. Figure 3.7 shows this indicator as a light-grey square in the background. Once the user has selected this square the user is asked whether the assumed fact is true, false or unknown according to the user's knowledge.[4]

The user can manage the facts that are used by the ERDS system by (un)selecting in a list of all facts known by the ERDS system (see Figure 3.8). The user can also choose to add a new fact to the ERDS system. These added facts are only available to that specific user and tagged as 'play' facts (see Figure 3.9).[5].

3.2 Data Sources

Signals are generated from indicators that are described in one or more data sources. One of the current parallel developments in the field of emerging risk detection systems is the ERI system of TNO. This system created in the FoodInformatics project[6] uses text-mining techniques to automatically parse multiple data sources. Several food safety specialists define one or more textmining queries to harvest relevant information. This initiative focuses on getting new facts and leaves it to the experts to interpret these facts. ERDS on the other hand focuses on using facts in combination with modeled expert knowledge and tries to

[4]Note that this functionality is a mock-up.

[5]This functionality is implemented as a mock-up.

[6]Food Informatics is a sub-project within the Virtual Lab e-Science project and funded by the Dutch Ministry of Economic Affairs and the Ministry of Agriculture, Nature and Food Quality (http://www.vl-e.nl)

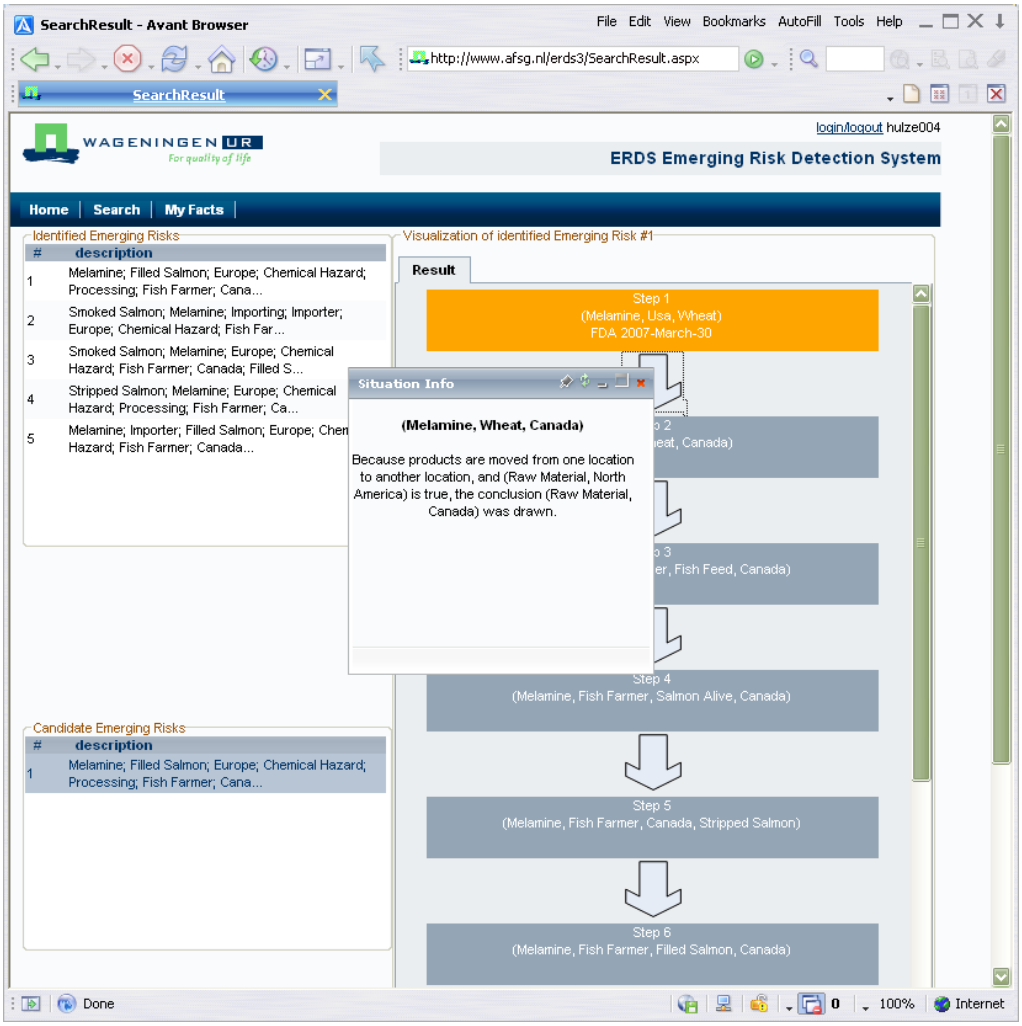

Figure 3.6: Zooming in on a risk path (explanation) (D)

find risk pathways. ERI and ERDS are therefore complementary systems (see Figure 3.10). Together with TNO we carried out several experiments in order to obtain from ERI new facts for ERDS by defining text-mining queries. The idea is that automatically new text-mined facts are added to ERDS. The results from the experiments indicate that ERI can be a useful source. Unfortunately this harvesting is too error-prone to be fully automated. Therefore we have the idea of collecting facts from ERI intermediated by a reviewer. Figure 3.11 shows a sketch on how such a reviewer can be supported. The reviewer selects the datasource that has harvasted all transportation facts from several data sources by text mining. The shown content below in the sketch shows the original sentence from which the fact was extracted. The column next to the sentence desribes the food type that is transported from one location (3rd column) to another (4th column). The data source is also mentioned in the next column. The reviewer has the ability to accept/reject the fact or change it. It turned out in the experiments that the from and to location are sometimes mixed-up. The reviewer can swap locations or edit parts of the fact. when the reviewer accepts the fact, the fact is added to the ERDS system. The depicted data in the sketch is real data collected from the experiment.

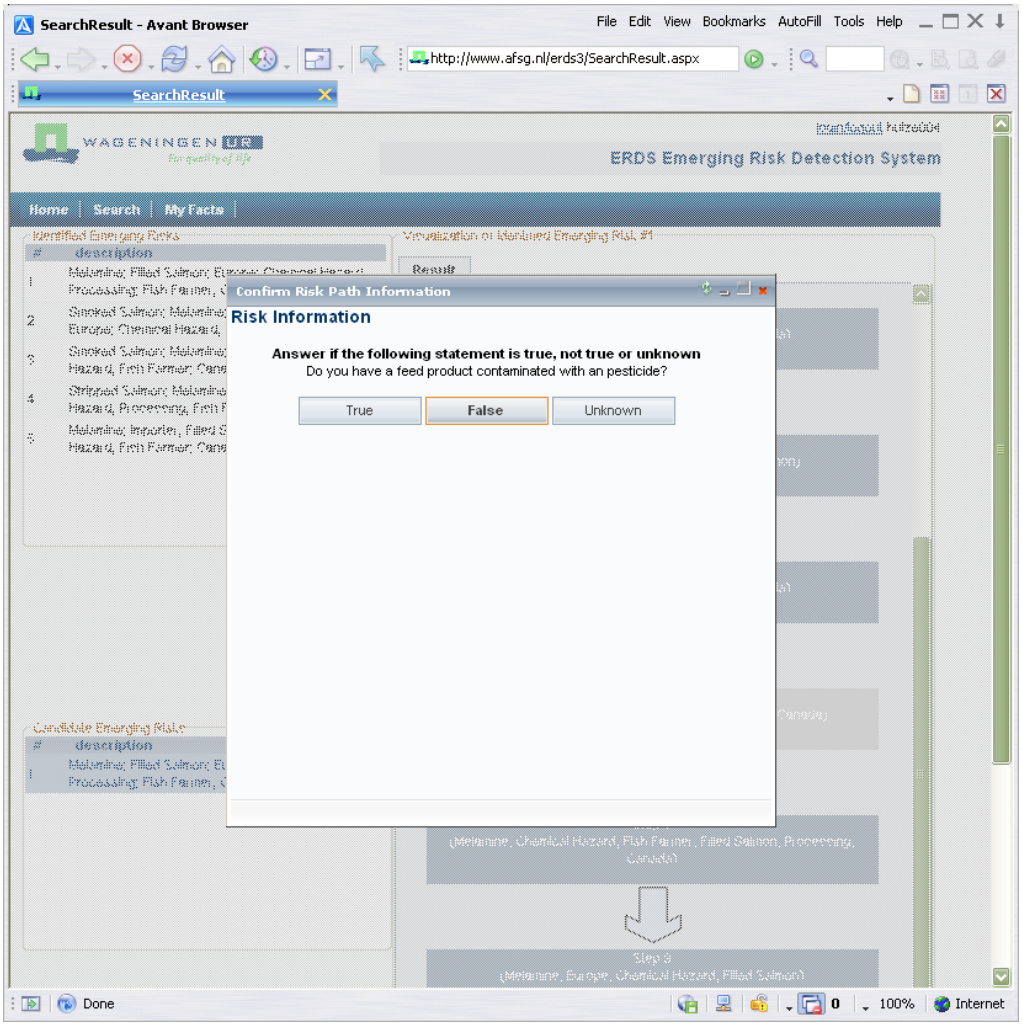

Figure 3.7: Filling in the missing facts (E)

3.3 Maintenance

The ERDS system is a system where its content is crucial for an accurate emerging risk detection. The latest signals and expert knowledge is needed to keep the system up-to-date. Therefore receives maintenance constant attention. Figure 3.13 describes the necessary maintenance taks, functionality and roles. We see two roles involved in the maintenance process:

- **The food safety expert**, He is using a fact editors and/or fact reviewers (like sketched in Figure 3.11 for adding and reviewing facts inside the ERDS system and newly harvested facts from external data sources. The food sagety expert is a real expert on the food safety domain, and is not especially skilled in IT and information modeling skills.
- **The knowledge translator**, maintaining the (complex) knowledge rules in the system, and (re)defining the facttypes. The knowledge translator is skilled in IT and information modeling and uses tools like Figure 4.1. Note that the knowledge translator is not a domain expert in the food safety domain. He always needs interaction and feedback from food safety experts.

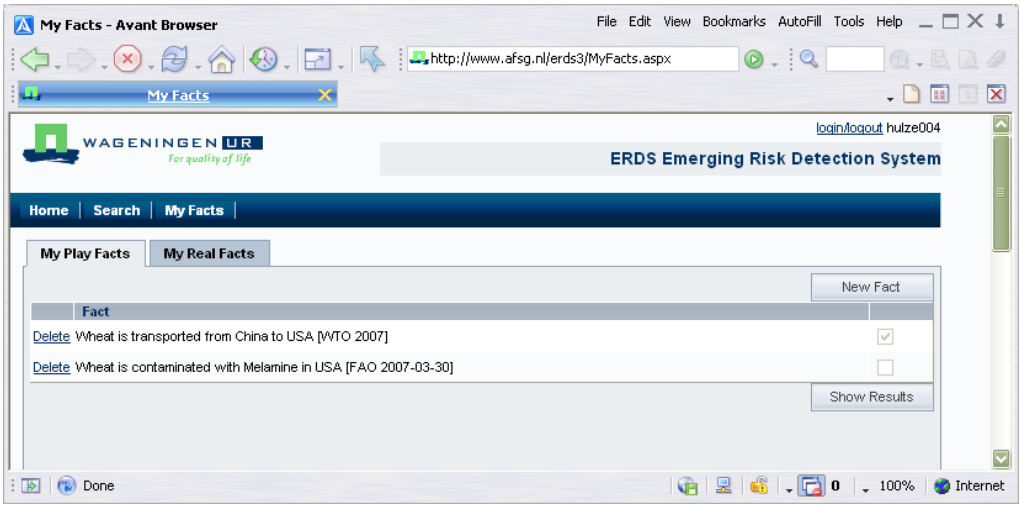

Figure 3.8: (un)select facts (F)

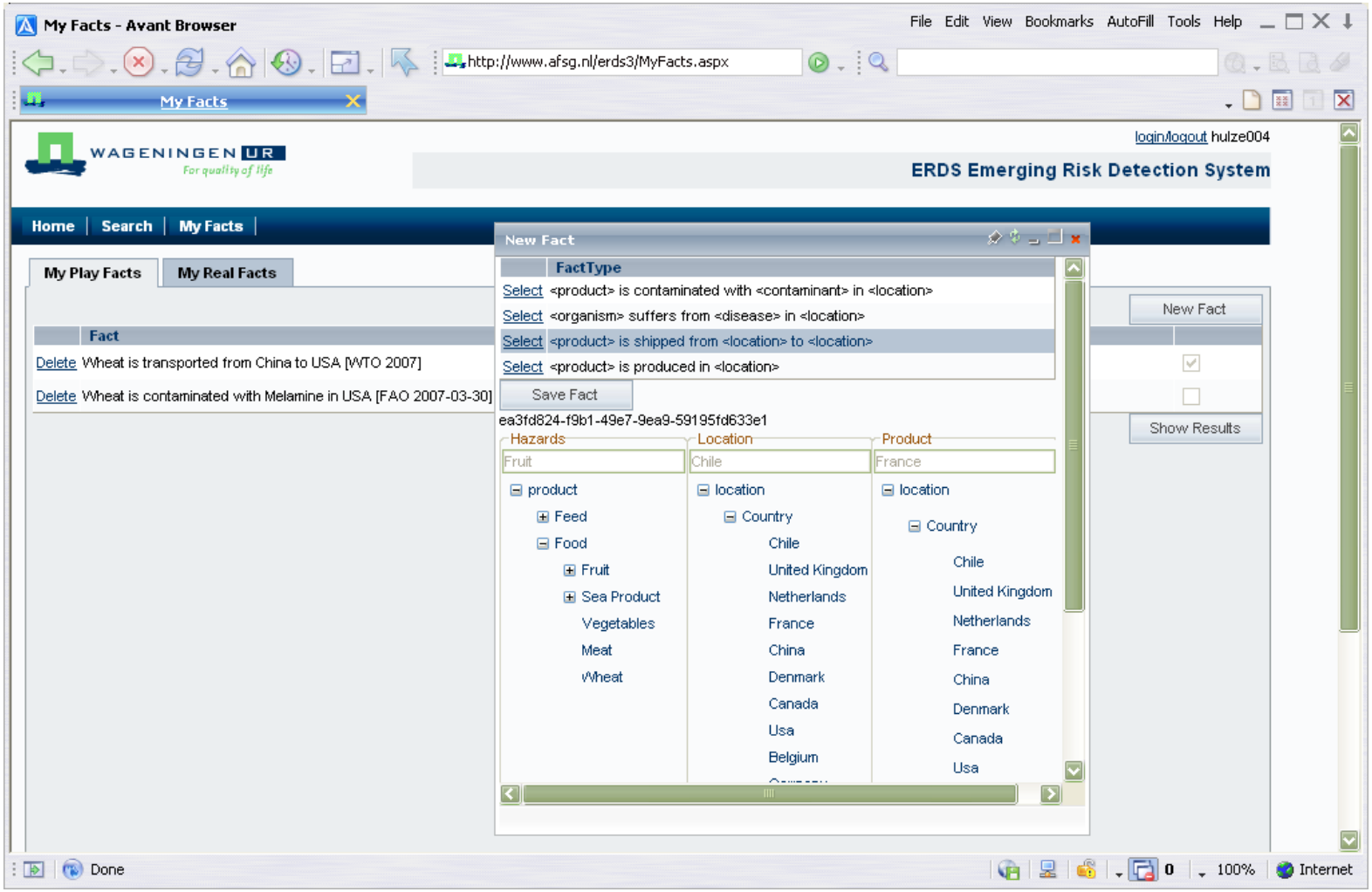

Figure 3.9: Adding new 'play' facts (My Facts) (G)

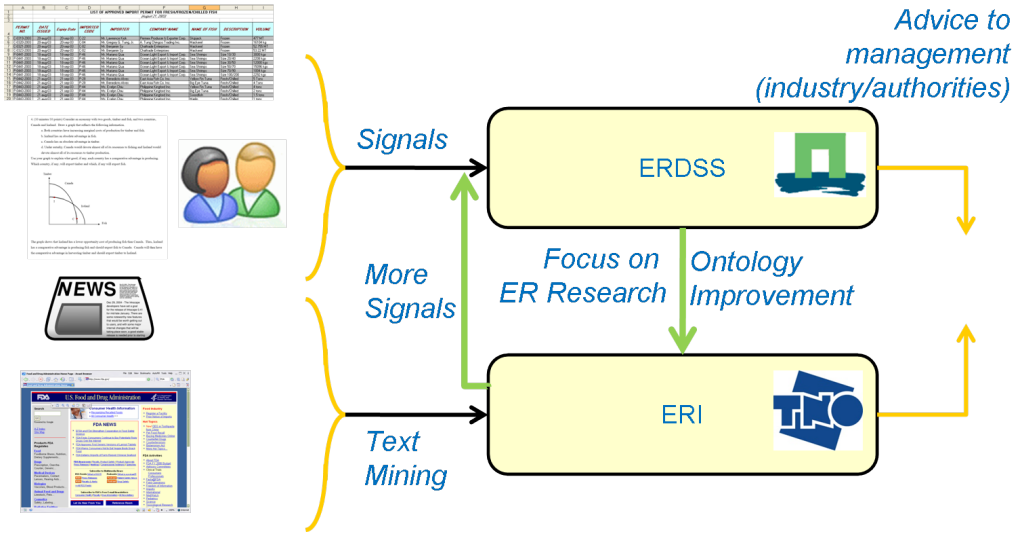

Figure 3.10: The link between ERDS and TNO's system ERI

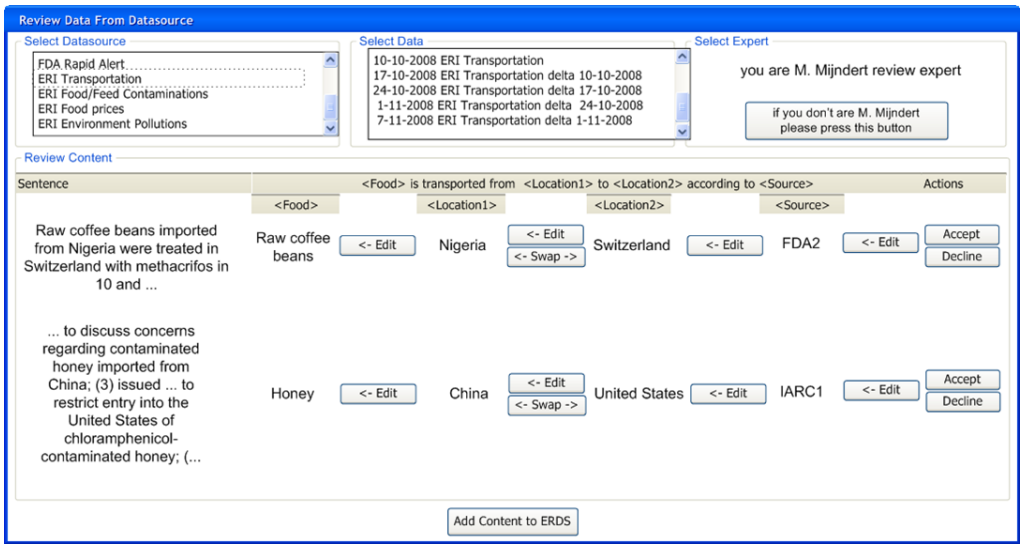

Figure 3.11: Harvesting facts from ERI into ERDS

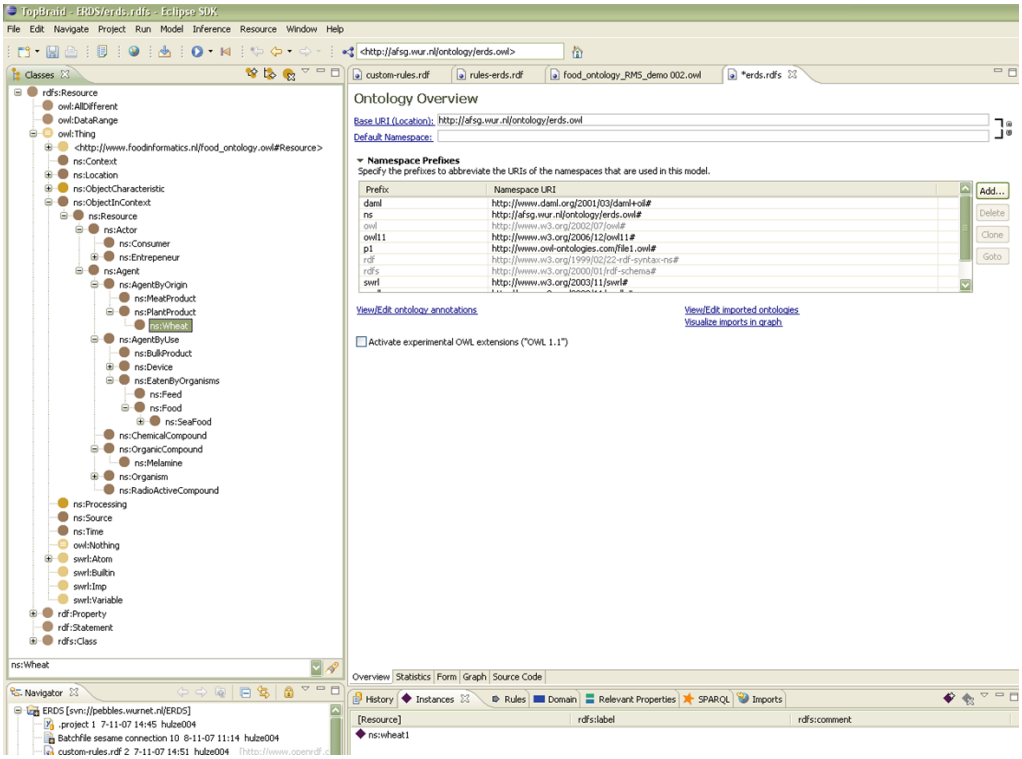

Figure 3.12: Screenshot of the ontology editor

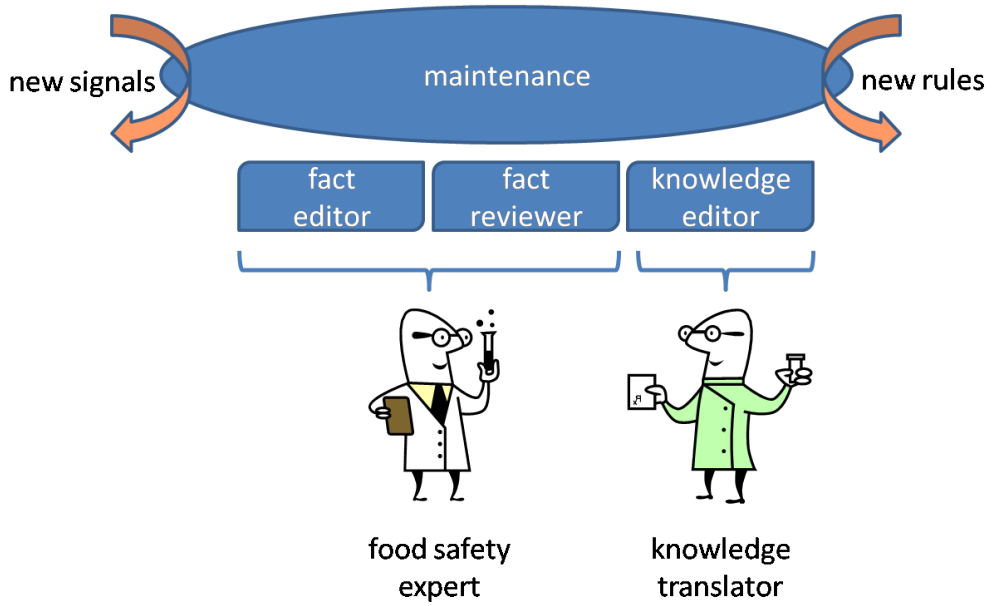

Figure 3.13: Maintenance tasks and roles

Chapter 4

Knowledge Modelling

4.1 Introduction

In this chapter we describe the structure in which we model the acquired facts. Prerequisites for this structure are the ease of understanding and encoding, expressiveness and capable to deal with differences in resolution (like facts on continent level, and facts regarding a local region). We looked to several formalisms that can express such a structure (also known as an ontology). The knowledge describing transformations of facts (reasoning) is described in the next chapter.

4.2 The knowledge structure (ontology)

As we described in previous chapters, detecting emerging risks is using the harvested facts from relevant data sources. The example fact 'Wheat is contaminated with Melamine in the USA according to the FDA site on 30 March 2007') has to be stored in a handy way for the algorithms to detect the emerging risks. Therefore we designed a knowledge structure (or ontology) that is a tree-like structure where all necessary entities and relationships between those entities are described.

To structure entities in a meaningful fashion, we introduce the modeling primitives of *class* and *instance*. Any concept that represents a group or set of individually distinguishable items is called a class. For example, the concept 'Country', which represents a group of actual countries, can be considered a class. Any member of a class is called an *instance* or individual, for example 'Canada' is an instance of the class 'Country'. Along with these two notions, we also introduce a relation between classes, the *subclass* relation. When some class A is specified to be a subclass of some class B, this is to be interpreted as meaning that all instances of class A are also instances of class B. For example, if we have a class 'Country' and a class 'GeographicEntity' and we assert that 'Country' is a subclass of 'GeographicEntity', the consequence of this assertion is that all instances of 'Country' (for example 'Canada') are also to be considered instances of 'GeographicEntity'. The hierarchical structure of our ontology is therefore formed by the tree-like structure built up from class concepts and the subclass relations between them.

Note that for some pairs of classes it is natural to relate them as subclasses (for example 'Country' and 'GeographicEntity') but for others it is not. For example, consider the classes 'Country' and 'Province'. At first glance it may seem useful to have 'Province' directly below 'Country' in the hierarchy of our knowledge structure. However, to do so would introduce a modeling error: after all, it is not generally true that any instance of 'Province' is also an instance of 'Country'. For example, 'Quebec' is a province (of Canada), but it is not a country itself. The relation between 'Province' and 'Country' is not a subclass relation but a *part-of* relation and therefore needs to be modeled differently.

We make two major distinctions on top-level: entities that describe the Context and entities that are in the context (ObjectInContext) (See Figure 4.1).

4.2.1 Context

The Context describes the time and space aspects of an object. Each new context (e.g. Context1 is desribed as an instance of 'Context'. Each fact is placed in this context describing the

- **location**. This can be Air ('stratosphere'), Land ('the Netherlands') or Water ('the North sea') related location[1]. The example fact has land location 'USA', and is stored as 'Context1' has relationship 'has location' with instance 'USA1' that is an instance of 'USA' that is a subclass of 'Country' that is a subclass of 'Location';
- **source**. Each fact is coming from a data source. This source is an indicator for the credibility and reliablilty. The example fact has source 'The FDA site' as is expressed as 'Context1' has relationship 'has source' with instance 'FDA1' that is an instance of Class 'FDA' that is a subclass of 'Source';
- **status**. A fact can have different states: like derived (as a result of transformations (rules)), 'real' (harvested in knowledge sources) or 'assumed' (used in what-if scenario's). In the example the status is 'real' and is expressed as 'Context1' has relationship 'has status' with value 'real';
- **time**. Relevant to know when a fact was issued from a datasource. The example case has time 30 March 2007' and is expressed as 'Context1' has relationshop 'has time' with value '30 March 2007'.

4.2.2 Object In Context

The ObjectInContext (from now on abbriviated with OIC) stores the remaining parts of the facts ('Wheat is contaminated with Melamine'). We devide the objects into two groups Actors containing all roles in production chains (like 'farmers') and Agents containing all (non)toxic stuffs like organic compounds. Between the objects in OIC are all sorts of relationships modeled. The example case is expressed in 'Wheat1' instance in class 'Wheat' subclass of 'Plant Product' subclass of 'Agent', 'Melamine1' instance of 'Melamine' subclass of 'organic compound' subclass of 'Agent'. 'Wheat1' has relationship 'is contaminated with' with instance 'Melamine1'. 'Wheat1' has relationship 'is in context' with instance 'Context1'.

4.3 Knowledge Representation with RDF

Formal knowledge representation in a form that is machine-interpretable (as is the goal in the ERDS system) requires that the defined ontology is specified in a format that can be processed by software tools in a way that closely resembles the intended semantics of the modeled information. The World Wide Web Consortium[2], in its Semantic Web effort, has defined standardized languages for precisely this goal. We will briefly introduce two of these formalisms here: RDF and RDF Schema.

4.3.1 RDF

The Resource Description Framework (RDF) [Klyne and Carroll, 2004] is a language for representing information about resources in the World Wide Web. It is particularly intended for representing metadata about Web resources, such as the title, author, and modification date of a Web page, copyright and licensing information about a Web document, or the availability schedule for some shared resource. However, by generalizing the concept of a "Web resource", RDF can also be used to represent information about things that can be identified on the Web, even when they cannot be directly retrieved on the Web. Examples include information about items available from on-line shopping facilities (e.g., information about specifications, prices, and availability), or the description of a Web user's preferences for information delivery.

RDF is based on the idea of identifying things using Web identifiers (called Uniform Resource Identifiers, or URIs), and describing resources in terms of simple properties and property values. This enables

[1] Notions of regions ('western europe') and communities ('the EU') are also incorporated

[2] See `http://www.w3.org/`

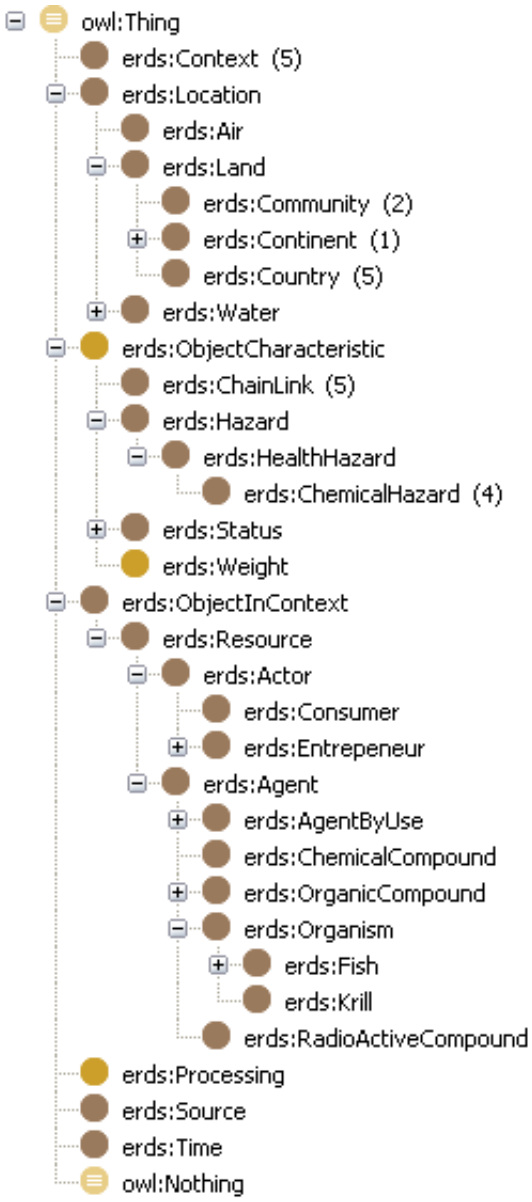

Figure 4.1: The ERDS Ontology (a fragment)

RDF to represent simple statements about resources as a graph of nodes and arcs representing the resources, and their properties and values. For example, we can represent the group of statements ”there is a person whose full name is John Doe, whose mail address is johndoe@example.org and whose phone number is 555-1234” as the graph depicted in figure 4.2.

4.3.2 RDF Schema

As we have seen in the previous section, RDF provides us with a basic (graph-based) model for knowledge representation but offers very little in ways of further structuring of such ’knowledge graphs’. To overcome this, RDF Schema has been developed.

RDF Schema [Brickley and Guha, 2004] is a Vocabulary Description language that extends the ba-

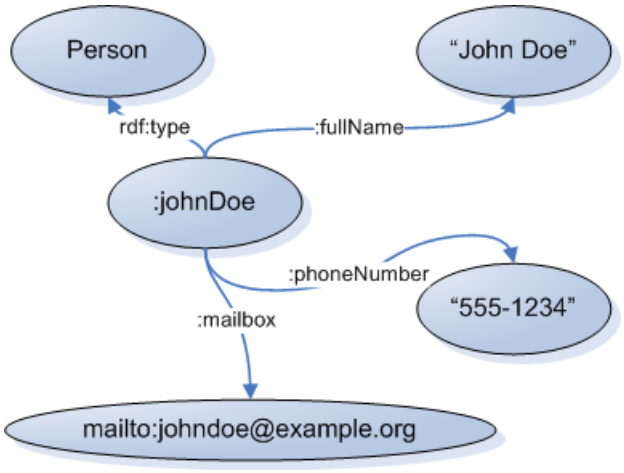

Figure 4.2: An example RDF graph

sic RDF model with additional modeling primitives. What this comes down to is that the RDF Schema definition introduces terms that can be used as property or concept names in an RDF graph, and *predefines* the meaning of these terms. Specifically, RDF Schema introduces terms such as `rdfs:Class` and `rdfs:subClassOf`, which can be used to define certain concepts as classes and introduce a class hierarchy between these classes.

As we have seen in section 4.2, the notions of class and subclass relations are key elements in our knowledge modeling, and RDF Schema provides us with a standardized set of primitives that we can use for this purpose. The advantage of using standardized terminology over simply creating our own terms and relations for this purpose is that of *interoperability*: by complying with an open standard we ensure that any database, query engine, or reasoning tool that implements the standard will be able to process our model.

Chapter 5

Reasoning Techniques

5.1 Introduction

The notion of *reasoning* is at the core of what the ERDS system is envisioned to do: ERDS will take existing pieces of information, and combine this information with background knowledge on processes and chains to produce new results (derived facts).

In this chapter, we will take a closer look at various reasoning formalisms and techniques for implementing these formalisms. We will also investigate various actual reasoner implementations and compare them in terms of performance and usability for the ERDS case.

5.2 Representing Rules

In the previous chapter we have seen how atomic facts, ontological knowledge about the ERDS domain, can be formally represented in an RDF Schema knowledge structure.

In ERDS however, a significant part of the relevant knowledge is not expressed as atomic facts, but rather as *rules*, or more precisely, *entailment rules*. For example, the knowledge that "if wheat is present in the USA then it also present in Canada" is an entailment rule: it defines that the presence of wheat in Canada is *entailed* by the presence of wheat in the USA, it is a *logical consequence*.

Rules typically follow the following pattern: `IF` *premise* `THEN` *conclusion*.

The *premise* of a rule consists of a set of atomic facts that are a necessary condition for the *conclusion* to be true. The *conclusion* itself also consists of atomic facts, which are the logical consequence of the rule, to be considered true after the rule has been evaluated and all the conditions in the premise have been found to hold.

In order for software tools to be able to use rules (interpret them, evaluate them), a syntax format needs to be chosen. Unfortunately, for rules, there is no globally accepted syntax format like there is for ontologies (although there are attempts to come up with a standard, there is actually very little tool support for those standards). However, an entailment rule shares many traits with a query: both rules and queries need to extract knowledge from a source of information (a knowledge base). Queries typically only extract information and then pass that information on (e.g. to the user who typed in the query), rules however use the extracted information to create new information. The IF-part (the premise) of a rule can therefore be considered a query. While no standard rule languages exist, standardized query languages do exist. This has lead to various reasoning tools reusing the syntax of, for example, the SPARQL query language [Prud'hommeaux and Seaborne, 2008] to write down rules in. This approach not only has the advantage of conforming to a standardized formal syntax but also that various existing tools support the evaluation of SPARQL queries: a reasoner tool can therefore very easily be implemented on top of a SPARQL engine, 'borrowing' its query evaluation functionality.

5.3 Evaluating Rules

Once rules have been formulated a strategy for evaluating them is needed. There are two basic strategies for rule evaluation which we will briefly discuss here: *forward-chaining* reasoning and *backward-chaining* reasoning.

In forward-chaining reasoning, we take the known fact base (the data) as our starting point. For each rule, we check if the premise is made true by the data, and if it is, we can add the conclusion of the rules to the fact base. In other words, forward-chaining reasoning follows the 'IF-THEN' rule from left to right, from premise to conclusion.

Backward-chaining reasoning works in the opposite direction: it first examines the conclusion of a rule and based on whether or not that conclusion is relevant to its goal (for example, the original question the user gave the system), it decides to check whether or not the premise holds.

In the chosen formalism for knowledge representation, RDF Schema, a standard proof system is present [Hayes, 2004], which consists of a set of entailment rules detailing such things as the consequences of subclass-inheritance and so on (for a full overview of the kind of reasoning that is supported by RDF Schema see [Hayes, 2004]). Many tools and frameworks for RDF Schema therefore provide tooling to deal with these entailment rules. In [Broekstra, 2005] and [Stuckenschmidt and Broekstra, 2005] various approaches for dealing with reasoning in RDF Schema are described in detail.

However, the kind of reasoning that is needed in the ERDS project is of a slightly different nature. While at least some of the knowledge that is available about our domain can be captured using RDFS modeling principles and its reasoning system, not all of it can: domain-specific rules like 'if a bulk product is present in a continent, then it is present in each country of a continent' are not expressible using only the simple modeling primitives RDF Schema provides. For this knowledge we need an extension to the RDF Schema formalism that allows us to specify this knowledge in the form of entailment rules, and a tool that will allow us to evaluate these rules and present the results in a practical, flexible, and scalable fashion.

5.4 Reasoning Tools

5.4.1 Relational Database Reasoning

We started with implementing a associative semantic reasoner in the relational database SQL server. This reasoner deals with situations, that represent a fact or signal, and consists of one or more related terms from the ontology. The situation 'wheat, melamine and USA' represent for instance the fact 'wheat in the USA is contaminated with melamine'. The advantage of this type of this implementation is a) that we could fully control the reasoning process and b) we deal with a highly informal way of expressing facts, leaving as much to the expert's interpretation in order to trigger additional knowledge and viewpoints of the expert.

However we discovered that the performance of the reasoner was poor. It was clear that a 'classical' relational database is not the optimal reasoning environment. The loosely-modeled facts had a disadvantage that it lead to ambiguity and unclear communication with the end user's of the system. That target audience is really accustomed to clear and unabigious information.

5.4.2 Ontology-Based Reasoning

As we have seen in chapter 4, we have chosen RDF Schema as appropriate formalisms for representing knowledge in the ERDS system. The reasons for choice have to do with the fact that knowledge-intensive applications like ERDS benefit from an explicit knowledge representation in a model such as RDFS, and moreover a good selection of tooling is available for handling information in such models, storing and querying large quantities of such data, and also using various reasoning techniques on top of that data.

In this section, we will look at several ontology-based reasoning tools that we consider relevant for the ERDS project.

The Custom Rule Reasoner

The Custom Rule Reasoner (CRR) is a reasoner implementation developed by AFSG, explicitly for use within the ERDS project. It is developed as a component of the Sesame Framework[1] [Broekstra et al., 2002].

The fact that this reasoner is a Sesame component is a tremendous advantage. Clearly, the ERDS project needs very scalable and flexible support for storage and retrieval of data. Given that we have chosen to model the ERDS data in RDF(S), the Sesame framework is an excellent choice for providing such scalable storage and retrieval: it has very good performance and scalability [Guo et al., 2005], is available under an Open Source license[2], and provides a flexible environment by design: it supports multiple database types and various way to plug in additional functionality as required. Wageningen UR has excellent in-house expertise on Sesame as well as close ties with Aduna, the company developing and distributing Sesame.

The CRR functions as a rule-based forward-chaining reasoner on top of a Sesame RDF repository. It reuses Sesame's own support for the SPARQL query language, effectively using SPARQL query syntax for the formulation of domain entailment rules.

The CRR in its current implementation uses a basic iterative entailment strategy, materializing the deductive closure on upload: while adding new data to the repository, the CRR entails all conclusions based on the set of rules and adds those conclusions to the repository as well. The advantage of this materialization approach is that it significantly simplifies query answering - every query answering task is reduced to simple uplook rather than on-demand rule evaluation. A potential disadvantage is the potentially large footprint of the storage of a full closure and the potential performance bottleneck in updating the repository (see a.o. [Broekstra, 2005]).

Networked Graphs

Networked Graphs [Schenk and Staab, 2008] are another declarative mechanism for use of SPARQL queries as rules. It can be seen as a view definition mechanism for RDF graphs.

The principle rests on the notion of *named graphs*: the capability of collections of RDF statements to be 'compartementalized' into named subcollections, so-called named graphs. The networked graphs approach extends this basic mechanism with a function that allows an *intensional definition* of the contents of a named graph. For example, in the ERDS case, if we choose to represent each country as a separate named graph, we can not only say 'the graph about Canada contains the fact that there is salmon being produced', but also we can say 'the graph about Canada contains the rule that if there is some contaminated bulk product in the USA, it is also present in Canada'. In other words, the contents of the graph about Canada is defined by (the consequences of) a rule rather than by explictly enumerating all the facts that should be in that graph.

An implementation of the Networked Graphs approach is the Networked Graphs Reasoner (NGS). This NGS is a reasoner component that, like the CRR we have seen in the previous section, operates on top of a Sesame RDF repository. A major difference with the CRR however is that the NGS operates in *backward-chaining* fashion: it does not precompute the deductive closure but only determines the view extensions that are required to answer a particular question, at question-time. The advantages of this approach are a relatively small additional demand on storage space and of course no additional performance burden during updates on the knowledge base. However, a potentially large disadvantage is the performance bottleneck in answering queries: after all, query answering involves on-demand reasoning rather than just simple lookup.

Euler

Euler[3] is an inference engine supporting logic based proofs, developed by Jos de Roo at Agfa Labs. It is a backward-chaining reasoner that can operate on top of RDF knowledge bases.

An attractive feature of Euler is that is has built-in support for proofs: not only can it deduct that a certain fact is true, it can also explain *why* it is true, that is, which rules and facts it used to arrive at its conclusion.

[1] see `http://www.openrdf.org/`
[2] See `http://www.openrdf.org/license.jsp`
[3] see `http://www.agfa.com/w3c/euler/`

A potential problem with deploying Euler in the ERDS system, though, is that the tool is primarily developed a standalone tool that is to be used directly from the command line. It is at this time not clear whether reuse as a component in a larger system is practically feasible.

5.5 A Comparison of Approaches

In order to determine which of the available modeling and reasoning techniques is most appropriate for ERDS, it is necessary to do a comparison of the techniques, in terms of various criteria. For quantitative criteria such as performance, we need a *benchmark* test, so that we can compare performance of the various aspects. For more qualitative criteria, such as ease of use, expressivity of the language, availability of tooling and so on, it is not possible to do an object performance test, but we can judge each approach on its relative merit for each qualitative aspect.

Setting up our own benchmark test is necessary for various reasons. First of all, very few general benchmark reports are available for the selection of tools and techniques in mind. Although the available literature does give an insight in general performance of at least some of these tools, they have not been performed on the ERDS data set. In order to be able to make a good comparison, it is necessary that we compare using (a part of) the ERDS knowledge base and rule base, as well as testing task performance in tasks that are actually relevant to ERDS (for example, answering particular kinds of queries). In the next sections, we will look in detail at how such a benchmark should be set up for ERDS.

5.5.1 Benchmark Criteria

As we have seen in previous sections, different reasoner implementations may have different advantages and disadvantages. This leads us to observe that in order to do a fair comparison between strategies we need to consider various performance criteria.

The main criteria we will base our comparison on are:

1. *query performance*: how quickly does the system provide an answer to a question.
2. *modification performance*: this criterium looks at the performance of the system when the knowledge base or the rule base is updated: how fast is this, how well does it deal with large vs. small updates, and so on.
3. *resource footprint*: the amount of disk space or memory resources the system consumes.
4. *scalability* : the overall performance of the system on ever-increasing sizes of data sets, rule bases, and so on.

5.5.2 Benchmark Data Set

In order to measure performance of the system we need a sample data set on which to perform our measurements. This sample data set needs to have the following characteristics:

1. *accurate reflection of the domain*: the data set needs to accurately reflect the ERDS domain in terms of use and complexitiy of the modeling constructs present. After all, if we were to measure performance on a completely different domain it would be difficult to predict how results obtained there are influenced by the presence or absense of certain knowledge modeling constructions.
   For this reason it is important that we conduct our own tests rather than rely solely on available benchmarking reports of various tools: while such reports give a general indication of performance, peculiarities of the ERDS domain may heavily influence the eventual performance of the system.
2. *sufficient size*: the data set needs to be of sufficient size to be able to obtain statistically significant performance differences between different approaches. Typically, query response times are measured in milliseconds (ms). If the sample data set is so small that every tool we look at can complete a query request within, say, 100 ms, we have no accurate measurement on which to decide (all the more since

typically PC time measurement has a granularity of about 10-30ms, making any difference of less than 30ms unreliable).
Moreover, it may well be that performance of various tools degrades worse than linearly as the size of the data set increases linearly.

5.5.3 Benchmark Queries

Once we have a data set and a set of criteria we need a set of tasks on which to measure performance. Since ERDS is essentially a query answering system, these tasks can be formulated as queries, for example: "Are there any chemical hazard predicted on bulk products in Europe?".

These queries need to have the following characteristics:

1. *accurate reflection of the domain*: the queries involved in tests need to closely reflect the envisioned real tasks of the system. Clearly, it would not do us any good to, for example, measure performance on a query that gives us the names of all kinds of salmon if that kind of query will never occur in practice.

2. *spread of general and specific queries*: the total set of queries needs to contain a good spread of various kinds of queries likely to occur in practice. Some systems may be very good at answering very specific queries ("every chemical hazard of type X on feed given to salmon from canada as occurring in the netherlands") but perform quite poorly on more general ones ("every hazard in europe"), vice versa. To be able to compare we need to have a set of queries that reflects this difference.

5.6 Current Status and Future Work

At the time of writing, we have identified the above criteria and have started exploring the ERDS domain for required data and queries. Additionally we are looking at a framework setup for doing accurate performance measurements.

We have also done some preliminary testing involving a few simple queries and a rather small data set, which we nonetheless believe to be representative of the ERDS domain in terms of expressivity. These tests involved only the CRR and the NGS system. While the data set was not of sufficient scale to merit definite conclusions, we have observed the following:

- the Custom Rule Reasoner has good query performance, and is quite robust against slightly different formulations of the same query. The NGS approach, however, seems to be brittle in this respect: if the query is formulated 'exactly right' (that is, sufficiently precise) it can perform very well, however query performance seems to degrade very drastically if the query is formulated slightly different from the optimum.

- in terms of speed of adding data, both systems performed satisfactorily. However, the sample data set only contained about 1,000 facts and is therefore not of sufficient size to make any accurate observations about upload performance.

We have also observed a qualitative difference between the two approaches: interms of ease of encoding, we have observed that formulating domain rules in the CRR's rule formalism was significantly easier than in the case of the NGS system. Especially the coupling between a rule and a specific named graph made rule formulation in the NGS approach hard. For example, in the NGS approach, certain general rules which hold over more than one country have to be repeated for each named graph that represents a country. In the CRR approach, such a rule only needs to be formulated once and can be trivially formulated in such a way as to be globally applicable.

Future work remains more comprehensive testing, using the benchmarking principles outlined above, and extend the set of tested tools to include Euler and any other possible candidates that we may yet identify.

Chapter 6

Conclusion

We have seen that in the food safety area, emerging risks detection is complementary to state-of-the-art food safety research and early warning detection. These three disciplines mutually benefit from each other: emerging risk detection needs knowledge from early warning detection and food safety research. In return emerging risk detection gives direction to blind spots in food safety research and helps updating early warning systems.

Emerging risk detection can really work, as we have shown with the realistic Melamine case in chapter 2: although the pet-feed chain knew the origin of the problem and took protective measures, it still was a threat to other chains, even food chains. Detecting emerging risks can give those chains that have a link to the problem time to take the appropriate measures.

Apart from food safety agencies we also observed that industry quality standardisation organisations may benefit from the use of an emerging risk system. Such organisations focus on the same mid- and long term goals for improving quality and dealing with emerging risks.

In the development of our Emerging Risk Detection prototype (ERDS), we have entered a new technical area with respect to harvesting new signals from external data sources. The experiments carried out with the text-mining system of TNO look promising. Such systems can be seen as a valuable data source for picking up new signals. These experiments give us more direction on a very important question: how to keep the system up-to-date. Two different roles with different skills as background and tools to support the maintenance tasks were identified, however still more future work on this matter needs to be done.

In chapter 4 we have shown how a formal knowledge structure, an ontology, can be used to represent signals and expert knowledge on food and chains, and how we can exploit such a structure to capture and process this information. We have introduced the standardized languages RDF and RDF Schema and have shown how the functionality that these languages provide closely fits with the requirements that we have. Clearly, a system such as ERDS can benefit enormously from the wealth of available tooling and standardized vocabularies and languages available in this area.

In chapter 5 we have identified the need for reasoning support in the ERDS system and have explored various tools and methods for achieving this. We have outlined conceptual and technical requirements and have introduced requirements for testing various implementations in the form of a benchmark. Future work includes the creation of a large sample data set, a set of representative benchmark queries, and execution of benchmark testing for all candidate reasoning tools.

Acknowledgements

We like to thank the steering committee Wim Ooms, Hub Noteborn, Rob Theelen and Geert Houben, the project coordinator Hans Marvin and the project members Joop van der Roest, Marnix Poelman, Rian Schelvis-Smit, Birgit de Vos, Jos Smit, Kees Booij, Gerie van der Heijden, Fátima Kreft and Willie van den Broek for their useful comments, knowledge and effort.

Bibliography

[Brickley and Guha, 2004] Brickley, D. and Guha, R. (2004). RDF Vocabulary Description Language 1.0: RDF Schema. Recommendation, W3C. See `http://www.w3.org/TR/rdf-schema/`.

[Broekstra, 2005] Broekstra, J. (2005). *Storage, Querying and Inferencing for Semantic Web Languages*. PhD Thesis, Vrije Universiteit. ISBN 90-9019-236-0.

[Broekstra et al., 2002] Broekstra, J., Kampman, A., and van Harmelen, F. (2002). Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In Horrocks, I. and Hendler, J., editors, *Proceedings of the first International Semantic Web Conference (ISWC 2002)*, number 2342 in Lecture Notes in Computer Science, pages 54–68, Sardinia, Italy. Springer Verlag, Heidelberg Germany. See also `http://www.openrdf.org/`.

[Guo et al., 2005] Guo, Y., Pan, Z., and Heflin, J. (2005). LUBM: A Benchmark for OWL Knowledge Base Systems. *Journal of Web Semantics*, 3(2):158 – 182. See also `http://swat.cse.lehigh.edu/projects/lubm/index.htm`.

[Hayes, 2004] Hayes, P. (2004). RDF Semantics. Recommendation, W3C. See `http://www.w3.org/TR/rdf-mt/`.

[Klyne and Carroll, 2004] Klyne, G. and Carroll, J. (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax. Recommendation, W3C. See `http://www.w3.org/TR/rdf-concepts/`.

[Marvin, 2008] Marvin, H. (2008). BO-08-002, Emerging Risks in de Nederlandse Voedselketen, Jaarrapportage 2007. Final, Wageningen UR.

[Noteborn, 2006] Noteborn, H. (2006). Forming a Global System for Identifying Food-Related Emerging Risks (EMRISK). EMRISK Final Report, VWA.

[Prud'hommeaux and Seaborne, 2008] Prud'hommeaux, E. and Seaborne, A. (2008). SPARQL Query Language for RDF. Recommendation, W3C. See `http://www.w3.org/TR/rdf-sparql-query/`.

[Schenk and Staab, 2008] Schenk, S. and Staab, S. (2008). Networked Graphs: A Declarative Mechanism for SPARQL Rules, SPARQL Views and RDF Data Integration on the Web. In *Proceedings of the 17th International World Wide Web Conference*, Bejing, China.

[Stuckenschmidt and Broekstra, 2005] Stuckenschmidt, H. and Broekstra, J. (2005). Time-Space Trade-Offs in Scaling Up RDF Schema Reasoning. In *Web Information Systems Engineering (WISE)*, number 3807 in Lecture Notes in Computer Science, WISE 2005 International Workshops, New York, USA. Springer Verlag, Heidelberg Germany.

Index